

Tales from the Crypto

A Look at Embedded Design Security

The bad guys are out to steal your design, your code, your tools, your customers, and anything else that isn't nailed down. Forget about trust, every designer needs to pay more attention to verification. Fortunately, you can use some silicon to help you keep your secrets safe.

It's interesting to contemplate how much of a person's net worth devolves to a few kilobytes of account information scattered around the ether. I keep thinking someday we'll all wake up and sign onto our accounts as usual only to find every financial asset we thought we owned was zeroed overnight. Easy come, but even easier to go, when all it takes is a "security lapse"—a *noxymoron* (i.e., two words that should go together)—and a few milliseconds to steal your bits.

Identity theft is a big deal, even for embedded apps. Consider the ongoing brouhaha over counterfeit consumables like laptop batteries and inkjet printer cartridges.^[1] There are arguments on both sides of the issue, but no argument the stakes are high.

If breaking into silicon can happen at the speed of light, the solution is more silicon that can stop the nefarious bit busters in their tracks. Let's take a look at some new silicon security guards from Atmel that you can deploy to harden your design against attack.

WHO GOES THERE?

I asked him if he thought Detroit would win the World Series. He said no, but they put up a bloody good fight. We pulled them out of the Jeep because the World Series had been between the Cardinals and Browns.^[2]—Robert Gravin, "The 3AD in the Battle of the Bulge"

A fundamental challenge for any security regime is knowing who you're talking to, and that's exactly the capability Atmel's new "Crypto-Authentication" chips bring to the party. In the case of the

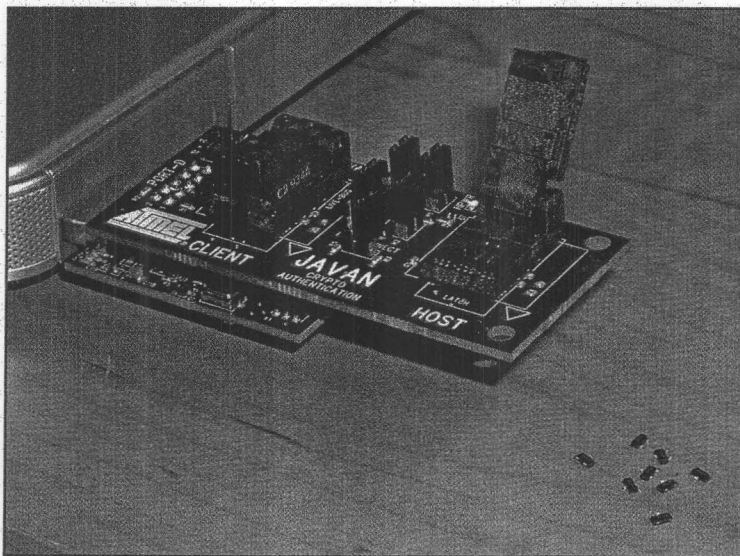


Photo 1—The JAVAN Crypto-Authentication starter kit makes it easy to put the '1025 and '10H5 through their paces, unless you get an attack of the drowsies. The fumble-fingered need not apply.

AT88SA102S, the protection comes at a remarkably small price. Not only is the per-chip cost low at just \$0.72 (in quantities of 100), but as you can see (barely) in Photo 1, board space for these tiny three-pin padlocks isn't an issue (although finding one you drop on the floor is). Running on anything from 2.5 to 5 V, the '102S does consume a healthy 10 mA during the time it takes to actively vet credentials. However, average power consumption will be much less, presuming the chip can spend most of its time off-duty sleeping at just 100 nA. Indeed, in many applications (such as the aforementioned batteries and inkjet cartridges), authentication may take place just once during initial installation.

The essence of authentication is that the host confirms that a would-be client knows a shared secret as in a typical password arrangement. You could configure a '102S to work that way and be done with it. With a whopping 256-bit "password," attempting to break-in by trying every combination will be slow going indeed. It's all the more true when you realize a brute-force attack is limited by '102S throughput to maybe 10 trials per second.

Hmm, with more than 10^{77} possible combinations, it will take more than 10^{76} seconds to try them all, which is a real long time, about a zillion years (OK, 10^{68}) plus or minus. But remember, chances are the crooks won't have to try every combination before stumbling on the right one. OK, so on average, it will only take half a zillion years. Do you feel lucky, punk? Well, do 'ya?

You'd think the evildoers would be better off stealing lottery tickets. However, there's an all too obvious weakness with a simple password scheme—namely, that the password can be overheard and replayed by an eavesdropper or "man in the middle." The spy-versus-spy solution calls for an elaborate scheme by which the spymaster (i.e., host) confirms that a to-be-vetted agent (i.e., client) knows a secret without requiring the agent to actually divulge the secret (in which case it might be overheard). If a

password is never passed, it can't be stolen.

Here's how it works with a '102S. The host issues a 256-bit "challenge" to the '102S. In turn, the '102S uses that number along with information hidden on-chip (e.g., a 256-bit "key," 48-bit unique chip ID, 64 bits of user-programmed "secret" fuses) as inputs to a "Secure Hashing Algorithm" (SHA-256).^[3] The 256-bit result of the SHA-256 calculation (called the "digest") is returned to the host in response to the challenge. Meanwhile,

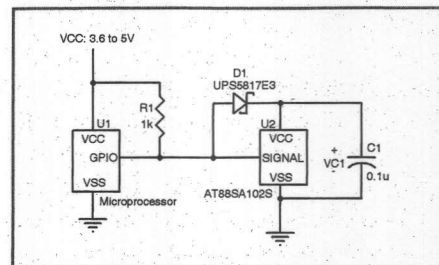


Figure 1—You can get by connecting a '102S with just two wires by tapping power from the signal line. Since the pull-up resistor (the '102S output is open drain) and bypass capacitor are required in any case, all it takes is the addition of a Schottky diode.

For 3

\$51 PCBs

FREE Layout Software!

FREE Schematic Software!

- 01 DOWNLOAD our free CAD software
- 02 DESIGN your two or four layer PC board
- 03 SEND us your design with just a click
- 04 RECEIVE top quality boards in just days

expresspcb.com

the host—knowing both the challenge it issued and the secret information that should be in an authentic client—can compare the expected result to that received from the '102S. If the digests match, the client is authenticated without the secret information ever passing hands.

Hashing is fundamentally a compression scheme that encodes a big—and possibly variable in length—chunk of data in a small piece of fixed length data. It's a concept that's fundamental to many aspects of computing. For instance, virtual memory and caching both rely on hashing to translate memory addresses from big spaces into smaller ones. A file system can use hashing to map, for example, an arbitrary customer name and address to a unique record number on disk. Heck, even the lowly CRC and checksum are examples of hashing functions that distill a packet down to a byte or two.

Security apps call for hashing functions with particular features to stymie would-be crackers. For instance, the chances that two different inputs to the hashing function will produce the same output (called

a "collision") must be minimized. The output of the hashing function should change in a seemingly random way, even for minor changes (i.e., a bit at a time) of the input. Oh yeah, it helps if the calculations involved don't require a supercomputer.

Far be it from me to weigh in on cryptographic nuances and controversies. I'll let Atmel speak for themselves when they say: "The NSA and universities involved in cryptographic research all agree that a brute force attack on SHA-256 is essentially impossible."^[4] At the same time, the algorithm (mainly a bunch of shifting and Boolean logic) is quite manageable (time and space) for even an 8-bit MCU.

Sounds good, but the scheme is still subject to eavesdropping if the challenge is always the same. However, by taking advantage of the user-programmed fuses (i.e., secrets) and unique chip ID, the jeopardy can be limited to a single host-client pairing. Eavesdropping on a '102S that works with a particular host won't help the bad guy break into another host that expects a '102S with a different chip ID or fuse secrets.

A more robust solution calls for the host to vary the challenge similar to the way your car door lock remote control uses a "rolling code." Now a transient eavesdropper is up a creek because the challenge changes each time and capturing a single challenge-and-response transaction does little good.

The simplest option has the host store a number of precomputed challenge and response pairings at 64 bytes per pair. The more the pairings, the longer it will take an eavesdropper to capture them all. A better solution—although it's one that requires the host to perform the SHA-256 calculation—factors a random number into the challenge, keeping in mind the usual caveats (i.e., how truly random your random number generator is).

It's clear a '102S can harden a design against brute-force and eavesdropping attacks, but what about an inside job? A possible weak link in the aforementioned scheme is that it depends on the security of the shared secrets stored in the host, secrets which may be stolen (e.g., source code), reverse-engineered (e.g., execution trace), or

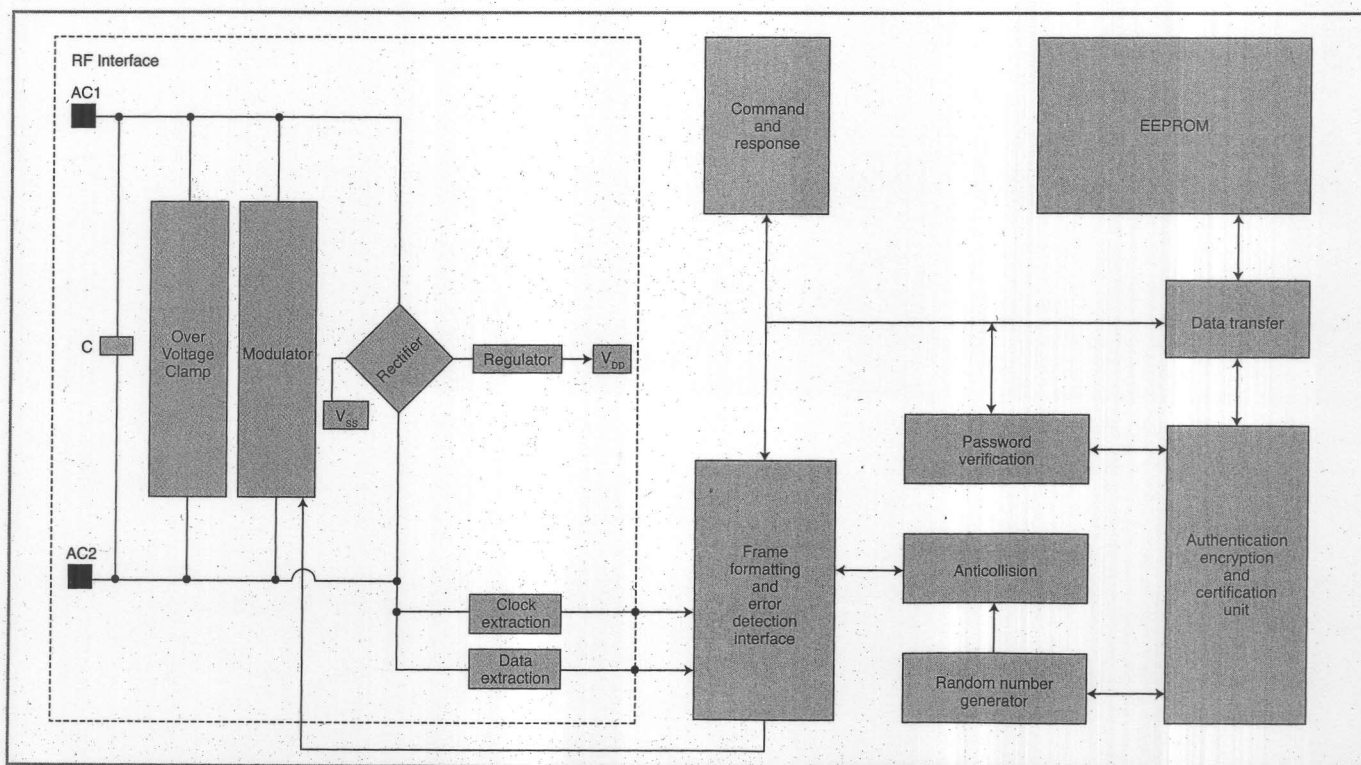


Figure 2—The RF04C is a passive RFID tag with a difference—namely, EEPROM to store your secrets and bulletproof security to keep them safe.

Command Name	Description
Abort	Exit command in progress
Clear	Exit command in progress, clear buffer, turn RF OFF
Poll Continuous	Poll continuously for Type B PICCs
Poll Single	Poll once for Type B PICCs
Read Buffer	Read data buffer
Read Register	Read configuration register
RF OFF	Turn off 13.56 MHz RF field
RF ON	Turn on 13.56 MHz RF field
Sleep	Activate standby mode
TX Data	Transmit data to PICC and receive the response
Write Buffer	Write data buffer
Write Register	Write configuration register

Table 1—I know little about RFID protocols and standards. And thanks to the intelligence embedded in the AT88RF1354 reader-on-a-chip, I never will. It hides the gory details—managing the radio, packet-handling collision resolution and so on—behind a simple serial interface and concise menu of high-level commands.

inadvertently exposed (e.g., software bug).

To that end, Atmel offers the AT88SA10HS, which is quite similar to—and designed to work in tandem with—the '102S. In this configuration, the host issues a challenge to both chips and delivers the response digest calculated by the '102S to the '10HS. In turn, the '10HS decides whether the '102S response is authentic and simply gives the host a yes/no result. Note that none of the secret information or results of any calculations are ever known (or knowable) by the host and thus can't be leaked.

That leaves a direct attack on the Atmel silicon—truly black-hat stuff—as the last resort. For obvious reasons, the documentation doesn't go into a lot of detail, but mention is made of a variety of physical protection mechanisms. These include chip shielding, tamper and pin-attack detection, internal clock dithering, and secrets dispersed across the die, not just sitting in a ROM.

LESS IS MORE

Needless to say, with just three pins (VCC, VSS, and SIGNAL), the hardware interface is easy. In fact, you can get by with just two pins using the old trick of tapping the I/O line to power the chip (see Figure 1). Atmel's application note titled "Crypto-Authentication" describes the details, such as how to size the capacitor and pull-up resistor considering the power supply (i.e. worst-case SIGNAL pin duty cycle) and demand (chip is awake, sleeping, calculating, performing I/O, etc.).^[4]

Having less pins to deal with may make the hardware easy, but it does require more work from your software driver. The single SIGNAL pin has to serve half-duplex duty in both directions, so the driver is responsible for

issuing commands to turn the bus around and avoid collisions.

Some applications might require the simultaneous authentication of multiple clients. The good news is that multiple '102S chips can ride on a single SIGNAL line. The bad news is that the protocol is a little goofy as it requires you to issue a command to each chip you don't want to talk to (telling it you don't want to talk to it) before you can talk to the chip you want to talk to.

Absent a pin for a clock, signaling is naturally asynchronous. The bit timing is a little bizarre in that it's asymmetric (i.e., 39 μ s from the host to the '102S versus 60 μ s in the other direction). I was figuring some bizarre bit-banging software would be required, but it turns out you can fake it with a standard UART running at 230.4 kbps. (Each byte trans-

ferred by the UART accomplishes a single bit transfer on the SIGNAL line.)

Chips without a RESET pin always make me nervous. It's almost inevitable that a hardware or software glitch will lead to a dead-end or loss of control that requires a power cycle. The '102S addresses the problem in two ways. First there's an I/O timeout that puts the chip to sleep if I/O gets out of sync. That way the chip won't get hung up waiting indefinitely for bits the host isn't sending. Similarly, a glitch on the SIGNAL line could wake up a sleeping '102S. Not knowing the '102S woke up, the host wouldn't know it needs to issue (or reissue) a SLEEP command. To solve this dilemma, the '102S has a watchdog timer that puts the chip to sleep a few seconds after it wakes up, no matter what (i.e., even if it is in the middle of executing a command or I/O). This strategy—which is,

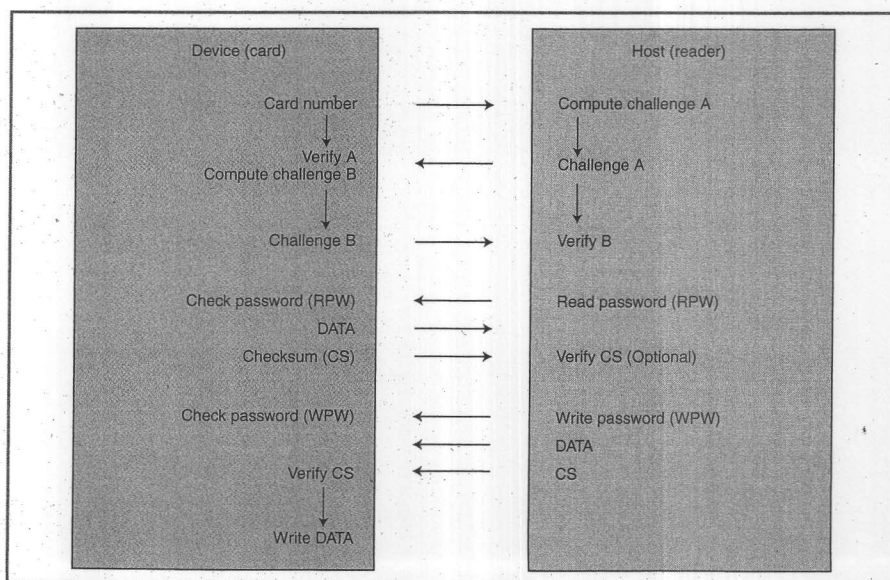


Figure 3—The RF04C features mutual (i.e., two-way) authentication in which both the host and client verify each other's identity and the integrity of messages is protected in both directions.

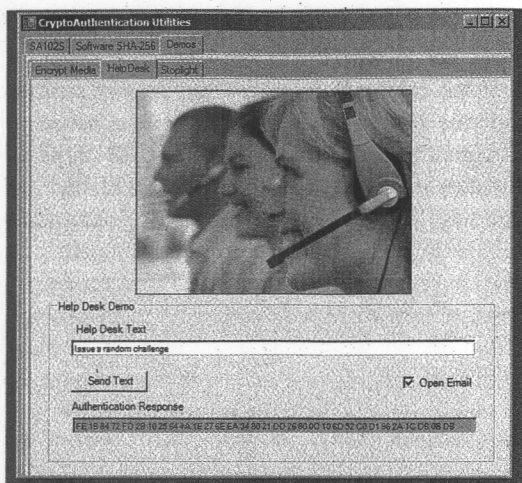


Photo 2—The JAVAN software exercises CryptoAuthentication chip features and also demos some example applications. Going beyond a password or “your mother’s maiden name,” the chips could be used to verify that a person on the other end of a phone call or e-mail truly holds the key.

“if in doubt, sleep”—is all well and good, but it means your driver code has to be cognizant of timing less it run afoul of the automatic failsafes.

It’s all a bit cumbersome, but maybe that’s appropriate. The bad guys will have to deal with this too. Why make it easier?

On the other hand, Atmel does just that with the JAVAN EV kit (\$99.95, quantity of one) that connects to your PC via USB. Thankfully, the board includes some cute little sockets, one for a “Client” (e.g., ‘102S) and one for a “Host” (e.g., ‘10HS). Otherwise, I’m just not sure how you would homebrew a programming and prototyping lash-up for these laughably tiny chips.

Better yet, the JAVAN kit comes with some handy PC utility software to access, program, and demo the chips (see Photo 2). Debugging security hardware and software can be tough because it’s just that—secure. There’s really no way to test that you’ve set up and programmed the Atmel chips correctly after the fact, other than trying them out. If you’re having problems, is it your chip setup or hardware

interface or driver software that’s to blame? Or maybe it’s all of the above? Having a stable and proven platform like JAVAN to start with really helps reduce hassles and head-scratching as you craft your own design.

TRICKY TAG TEAM

The CryptoAuthentication chips aren’t Atmel’s first foray into security. Earlier offerings included a line of CryptoMemory chips that I covered back in 2007 (“Silicon Secrets,” *Circuit Cellar* 205). You’ll recall the CryptoMemory chips combine security features (e.g., keys, passwords, authentication, and encryption) and defenses

against attack (e.g., password/authentication attempt counters, tamper detection, silicon shielding, and secret obfuscation) with 4 to 64 Kb of EEPROM.

Now Atmel offers “CryptoRF” parts that extend the CryptoMemory concept with the addition of RFID capability. Consider the AT88RF04C (\$0.95, quantity 100), which combines a 4-Kb CryptoMemory and a standard

(ISO/IEC 14443 Type B) 13.56-MHz RFID interface (see Figure 2). I’m not an expert in the RFID field, but it appears this “13.56-MHz Type B” is a very widely used standard, which means the ‘RF04C should be compatible with a lot of different readers. Or, for that matter, it’s easy to design your own since Atmel is also introducing the AT88RF1354, virtually a complete reader on a chip (\$1.59, quantity 100). You can put the pair through their paces with the “Bamboo” CryptoRF evaluation kit for just \$99.95 (see Photo 3).

The ‘RF04C is a so-called “passive” tag. That means it’s completely powered from the RF field generated by the reader, so the tag itself doesn’t need a battery and will keep working indefinitely. And despite their fragile appearance, the tags are robust enough to use across the industrial temperature range of -40° to 85°C , the only caveat being EEPROM data retention is derated at higher temperatures (e.g., 10 years at 55°C versus 30 years at 35°C). Write endurance is 100,000 cycles which seems more than adequate, even for long-life applications.

Combining “passive” power via RF with EEPROM poses unique challenges. Foremost among them is the fact that power may disappear at any time, possibly during writes to the EEPROM, which is a real no-no. To protect the integrity of EEPROM, the ‘RF04C incorporates a unique two-stage “anti-tearing” write cycle. When enabled, EEPROM write data is first sent to a temporary EEPROM buffer and then, after that write completes successfully, on to the main array. If power fails during the first (buffer) write, the main array data remains unaffected. If power fails during the second (main array) write, the chip tries again (i.e., writes the main array from the buffer) at the next power-up.

RFID is trickier than its lowly bar code and price-tag

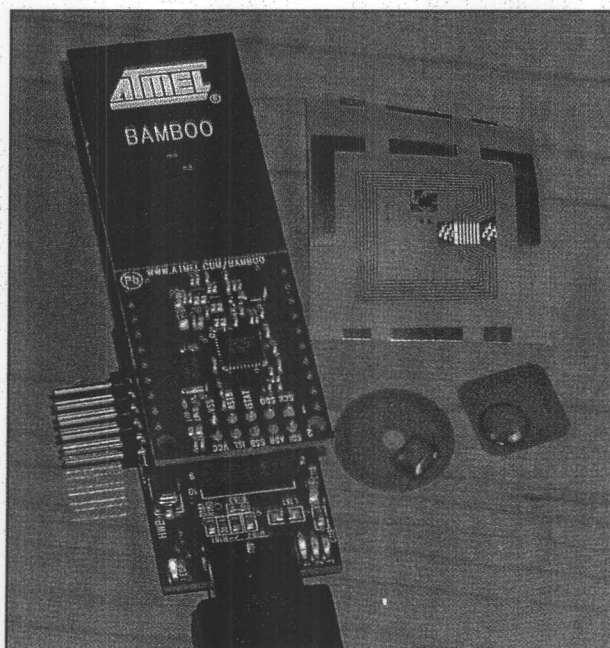


Photo 3—The BAMBOO CryptoRF starter kit comes with an USB plug-in AT88RF1354-based RF reader and quiver of RF04C CryptoRF tags.

